```
// Computer Program Listing Appendix Under 37 CFR 1.52(e)
// ParamManager.java
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.
package com.sybase.jdbc2.jdbc;
// ... (code omitted)
public class ParamManager implements Cacheable
{
    // ... (code omitted)
    // Constructor for light-weight copy.
    public ParamManager(ParamManager copyFrom, SybStatement stmt)
        throws SQLException
    {
        _next = -1;
        _last = -1;
        _maxOutParam = -1;
        _stmt = stmt;
        _context = stmt._context;
        _stmtMgr = stmt._statementManager;
        _protocol = _context._protocol;
        int numParams = copyFrom._params.length;
        _params = _protocol.paramArray(_context, numParams);
        for (int i = 0; i < numParams; i++)
        {
            _params[i]._paramMarkerOffset = copyFrom._params[i]._paramMarkerOffset;
        }
        _mgr = _context._conn.getSharedCacheManager();
        if (_mgr == null)
        {
            _mgr = new CacheManager(_context._is);
            boolean reReadable = _context._conn._props.getBoolean(SybProperty.REPEAT_READ);
            _mgr.setReReadable(reReadable);
            int cacheSize = _context._conn._props.getInteger(SybProperty.STREAM_CACHE_SIZE);
            if (reReadable)
            {
                cacheSize = CacheManager.INFINITE_CACHING;
            }
            _mgr.setCacheSize(cacheSize);
            // ?? max column size
            _mgr.setChunkSize(Const.COLUMN_CHUNK_SIZE);
            _mgr.setAbortOnCacheOverflow(true);
            _context._conn.setSharedCacheManager(_mgr); // register for re-use
        }
        if (copyFrom._templateHoldsParsedNoLiteralQuery)
        {
            _hasLiteralParam = copyFrom._hasLiteralParam;
            _savedParsedQuery = copyFrom._savedParsedQuery;
            _paramMarkersHaveBeenParsed = true;
        }
        _copiedFrom = copyFrom;
    }
}
// SybCallableStatement.java
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.
package com.sybase.jdbc2.jdbc;
// ... (code omitted)
public class SybCallableStatement extends SybPreparedStatement
    implements com.sybase.jdbcx.SybCallableStatement
{
    // ... (code omitted)
    // another constructor which is used only by the SybCallableStatementCopy
    // subclass. This constructor avoids unnecessary re-parsing of the query
    // string, and instead clones the necessary non-immutable sub-objects.
    SybCallableStatement(ProtocolContext pc, SybCallableStatement copyFrom)
        throws SQLException
    {
        super(pc, copyFrom);
        if (Const.DEBUG) Debug.println(this, "Constructor('" + _query + "')");
        _allowsOutputParms = copyFrom._allowsOutputParms;
        _rpcName = copyFrom._rpcName;
        _hasReturn = copyFrom._hasReturn;
        if (_hasReturn)
        {
            _paramMgr.setParam(1, Param.STATUS_RETURN, new Integer(0), 0);
            _paramMgr.registerParam(1, Types.INTEGER, 0);
            _returnHasBeenRegistered = false;
        }
    }
}
// SybCallableStatementCopy.java
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.
//
// Confidential property of Sybase, Inc.
// (c) Copyright Sybase, Inc. 2002.
// All rights reserved
//
package com.sybase.jdbc2.jdbc;
import com.sybase.jdbc2.utils.Debug;
import java.sql.*;
import java.io.IOException;
/**
* This class provides a light-weight way to share callable statements
* across connections.  Use the
* <PRE>
* public SybCallableStatement SybConnection.copyCallableStatement
*      (SybCallableStatement stmt)
* </PRE>
* Sybase extension to create these objects. That method produces a
* SybCallableStatement that is equivalent to the original one except that
* it is attached to the provided connection. The copying process is
* accelerated by the use of a "shared" ProtocolContext from that connection.
* <P>These shared prepared statement objects should only be used 1-at-a-time
* on a connection, and you should always call the close() method immediately
* after executing them.
* @see SybConnection.getSharedProtocolContext
* @see SybConnection.copyCallableStatement
*/
public class SybCallableStatementCopy extends SybCallableStatement
{
    // Constructors
    SybCallableStatementCopy(ProtocolContext pc, SybCallableStatement copyFrom)
        throws SQLException
    {
        //* DONE
        super(pc, copyFrom);
    }
    // A secret method to allow a statement/resultset/etc have a utility
    // statement on the same context.
    // Note that this method is called only by TdsCursor for language
    // cursor processing. Since the *StatementCopy classes are not to
    // be used with cursors (we mention this in the javadocs), this method
    // should not be called.
    public void switchContext(ProtocolContext pc)
    {
        if (Const.ASSERT) Debug.assert(this, false);
    }
    // override the close method to make sure we retain the shared context
    public void close() throws SQLException
    {
        close (false);
    }
}
// SybConnection.java
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.
package com.sybase.jdbc2.jdbc;
// ... (code omitted)
public class SybConnection implements com.sybase.jdbcx.SybConnection
{
    // ... (code omitted)
    private ProtocolContext _sharedPc;
    private CacheManager _sharedCm;
    // ... (code omitted)
    /**
     ** Use this method to create a light-weight copy of a PreparedStatement
     ** that may have been created on a different connection.
     **
     ** Originally intended as an internal hook for use by EAServer's CMP driver
     ** wrapper - com.sybase.ejb.cmp.SybaseConnection - to implement a prepared
     ** statement cache with very low memory and CPU requirements.
     **/
    public com.sybase.jdbcx.SybPreparedStatement copyPreparedStatement
    (com.sybase.jdbcx.SybPreparedStatement stmt) throws SQLException
{
    return (com.sybase.jdbcx.SybPreparedStatement)
        (new SybPreparedStatementCopy(getSharedProtocolContext(),
        (com.sybase.jdbc2.jdbc.SybPreparedStatement) stmt));
}
/**
 ** Use this method to create a light-weight copy of a CallableStatement
 ** that may have been created on a different connection.
 **
 ** Originally intended as an internal hook for use by EAServer's CMP driver
 ** wrapper - com.sybase.ejb.cmp.SybaseConnection - to implement a prepared
 ** statement cache with very low memory and CPU requirements.
 **/
public com.sybase.jdbcx.SybCallableStatement copyCallableStatement
    (com.sybase.jdbcx.SybCallableStatement stmt) throws SQLException
{
    return (com.sybase.jdbcx.SybCallableStatement)
        (new SybCallableStatementCopy(getSharedProtocolContext(),
        (com.sybase.jdbc2.jdbc.SybCallableStatement) stmt));
}
/**
 ** Get the shared ProtocolContext.
 ** MUST be used only by Syb(Prepared/Callable)StatementCopy.
 **/
protected synchronized ProtocolContext getSharedProtocolContext()
    throws SQLException
{
    if (_sharedPc == null)
    {
        _sharedPc = initProtocol();
    }
    return _sharedPc;
}
/**
 ** Get the shared Cache Manager.
 ** MUST be used only by Syb(Prepared/Callable)StatementCopy.
 **/
protected synchronized CacheManager getSharedCacheManager()
{
    return _sharedCm;
}
/**
 ** Set the shared CacheManager.
 ** MUST be used only by Syb(Prepared/Callable)StatementCopy.
 **/
protected synchronized void setSharedCacheManager(CacheManager cm)
{
    _sharedCm = cm;
}
```

```
}
// SybPreparedStatement.java
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.
package com.sybase.jdbc2.jdbc;
// ... (code omitted)
public class SybPreparedStatement extends SybStatement
    implements com.sybase.jdbcx.SybPreparedStatement
{
    // ... (code omitted)
    // another constructor for use by SybPreparedStatementCopy. This avoids
    // unnecessary re-parsing of the query string, and instead clones
    // the necessary non-immutable sub-objects.
    SybPreparedStatement(ProtocolContext pc, SybPreparedStatement copyFrom)
        throws SQLException
    {
        super(pc);
        _query = copyFrom._query;
        if (Const.DEBUG) Debug.println(this, "Constructor('" + _query + "')");
        _paramMgr = new ParamManager(copyFrom._paramMgr, this);
    }
}
// SybPreparedStatementCopy.java
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.
//
// Confidential property of Sybase, Inc.
// (c) Copyright Sybase, Inc. 2002.
// All rights reserved
//
package com.sybase.jdbc2.jdbc;
import com.sybase.jdbc2.utils.Debug;
import java.sql.*;
import java.io.IOException;
/**
* This class provides a light-weight way to share prepared statements
* across connections.  Use the
* <PRE>
* public SybPreparedStatement SybConnection.copyPreparedStatement
*    (SybPreparedStatement stmt)
* </PRE>
* Sybase extension to create these objects. That method produces a
* SybPreparedStatement that is equivalent to the original one except that
* it is attached to the provided connection. The copying process is
* accelerated by the use of a "shared" ProtocolContext from that connection.
* <P>These shared prepared statement objects should only be used 1-at-a-time
* on a connection, and you should always call the close() method immediately
* after executing them.
* @see SybConnection.getSharedProtocolContext
* @see SybConnection.copyPreparedStatement
*/
public class SybPreparedStatementCopy extends SybPreparedStatement
```

```
{
    // Constructors
    SybPreparedStatementCopy(ProtocolContext pc, SybPreparedStatement copyFrom)
        throws SQLException
    {
        //* DONE
        super(pc, copyFrom);
    }
    // A secret method to allow a statement/resultset/etc have a utility
    // statement on the same context.
    // Note that this method is called only by TdsCursor for language
    // cursor processing. Since the *StatementCopy classes are not to
    // be used with cursors (we mention this in the javadocs), this method
    // should not be called.
    public void switchContext(ProtocolContext pc)
    {
        if (Const.ASSERT) Debug.assert(this, false);
    }
    // override the close method to make sure we retain the shared context
    public void close() throws SQLException
    {
        //* DONE
        close (false);
    }
}
```